UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

Sistema de verificación de documentos usando árboles de Merkle

Aitor Muñoz Cuña
Tutor: Eloy Anguiano Rey

Septiembre 2018

Sistema de verificación de documentos usando árboles de Merkle

AUTOR: Aitor Muñoz Cuña
TUTOR: Eloy Anguiano Rey

Dpto. Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
septiembre de 2018

Resumen

Este Trabajo Fin de Grado trata de la utilización de las cadenas de bloques, o Blockchain, para enviar transacciones de forma segura e íntegra. Uno de los problemas de este sistema su lentitud, en http://www.blockchain.info se muestran tiempos de espera promedio para confirmar las transacciones, estos oscilan entre los 6 y los 11 minutos. Para solucionar esto, utilizamos Árboles de Merkle. Crearemos un bloque con toda la información, el cuál será metido en la cadena de bloques, de tal manera, que en una transacción tendremos la información de muchas. Asimismo, se reutiliza, no sólo el concepto de cadena de bloques, sino la red completa que usa Bitcoin para implementar el sistema descrito anteriormente.

Se ha desarrollado una web donde el usuario podrá indicar su clave pública, la cuál se utilizará cuando el usuario pida que encriptemos el archivo que envié. Asimismo, el usuario puede decidir si subir el archivo ya encriptado por él, o, si desea que el archivo se quede guardado en nuestra base de datos.

Estos archivos encriptados se almacenarán en una lista que, al finalizar el día, se introducirán en un árbol de Merkel para crear un bloque que se intentará unir a la cadena de bloques. Si el bloque es aceptado, la lista se vaciará para el día siguiente y se guardará en la base de datos un registro con el bloque que fue subido. Sin embargo, si el bloque no se queda en la cadena, la lista se guardará para el día siguiente con los nuevos archivos que entren.

Esto permitirá poder enviar transacciones de una forma más segura y barata, ya que no creamos un bloque por archivo. Igualmente, al ir encriptadas con las claves públicas, nos aseguraremos de que solo el destinatario pueda leer el archivo que le corresponde.

Abstract (English)

This Final Project deals with the use of block chains, or Blockchain, to send transactions in a safe and complete manner. One of the problems of this system is its slowness, at http: //www.blockchain.info average waiting times are shown to confirm the transactions, these range between 6 and 11 minutes. To solve this, we use Merkle Trees. We will create a block with all the information, which will be put into the chain of blocks, in such a way, that in one transaction we will have the information of many. Likewise, not only the concept of block chain is reused, but also the complete network that Bitcoin uses to implement the system described above.

A web has been developed where the user can indicate his public key, which will be used when the user asks us to encrypt the file that we sent. Also, the user can decide whether to upload the file already encrypted by him, or, if he wants the file to be saved in our database. These encrypted files will be stored in a list that, at the end of the day, will be inserted into a Merkel tree to create a block that will try to join the chain of blocks. If the block is accepted, the list will be emptied for the next day and a record with the block that was uploaded will be saved in the database. However, if the block does not stay in the chain, the list will be saved for the next day with the new files that enter.

This will allow to send transactions in a safer and cheaper way, since we do not create a block per file. Also, by going encrypted with public keys, we will make sure that only the recipient can read the corresponding file.

Palabras clave

MySQL, Blockchain, Árbol de Merkel, Bitcoin, AddRoundKey, AES, RSA.

Agradecimientos

A mi tutor Eloy Anguiano Rey, por darme la oportunidad de desarrollar un Trabajo Fin de Grado tan interesante y divertido. Asimismo, me gustaría agradecer su ayuda constante y seguimiento del trabajo.

A mis compañeros de la Universidad, por los consejos y experiencias que me han servido para mejorar este proyecto.

A mi familia, por estar ahí en todo momento y ayudarme a que me esfuerce cada día.

A mi pareja, por estar siempre a mi lado y ayudarme cuando lo necesitaba.

A Miguel por darme mi primer trabajo como socorrista que me permitió poder pagarme la carrera.

Aitor Muñoz Cuña

Junio 2019

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE FRAGMENTOS DE CÓDIGO

1 Introducción

1.1 Motivación

Estamos en una época en dónde la tecnología cada vez alcanza más protagonismo en nuestras vidas. Esto lo podemos ver en sanidad, en educación e, incluso, en muchas gestiones que tenemos que hacer en nuestro día a día. La importancia de estos datos y la forma en cómo los transferimos sin que puedan ser modificados, es uno de los principales motivos de este Trabajo de Fin de Grado.

Blockchain es el sistema de transferencia de archivos más innovador hoy en día, ya que nos permite realizar, sin intermediarios, transferencias fiables y seguras entre distintas personas. Asimismo, este método garantiza validez, inalterabilidad e inmutabilidad en los registros de las transacciones. [1]

Debido a su gran capacidad de registro, este sistema está cualificado para registrar virtualmente infinidad de documentos. Actualmente, se utiliza para registrar y autentificar, entre otros, títulos universitarios. No obstante, se está investigando un futuro uso en el área sanitaria, ya que este método permite ensamblar hospitales, aseguradoras y prestamistas, agilizando el proceso de forma exponencial. [1]

Principalmente, este método se emplea para criptomonedas, transacciones y sistemas de pago, cadena de suministros, contratos inteligentes y aplicaciones descentralizadas (Dapps), entretenimiento, comercio y registro de documentos. [1] Este último uso es en el que nos centraremos para llevar a cabo este Trabajo Fin de Grado.

Cada vez más expertos consideran que Blockchain será capaz de avivar un cambio digital que transformará, en un futuro, el mundo por completo. Sin embargo, este sistema es muy novedoso, por lo que todavía no se han desenmascarados todas sus funciones y, lamentablemente, hay inquietudes sobre su éxito a largo plazo. [7]

1.2 Objetivos

Desarrollar un sistema que permita, de manera segura, agregar una cuantiosa cantidad de archivos por bloque con el fin de devaluar el coste de cada transacción.

Implementar árboles de Merkel para lograr que todos los archivos que se reciban en un día se envíen como un solo bloque, lo que produciría un ahorro de recursos notable.

Añadir más seguridad al sistema, utilizando la clave pública para encriptar la clave simétrica, la cual se encarga, a su vez, de encriptar los documentos.

Diseñar una página web dónde el usuario pueda, mediante opciones, enviar los archivos al bloque, encriptar el archivo y enviarlo al bloque, guardar el archivo y enviarlo al bloque o encriptar y guardar el archivo y enviarlo al bloque.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Capítulo 2. Estado del Arte**

Estudiaremos diferentes funciones del Blockchain, similares a las aplicadas en este Trabajo Fin de Grado, que existen en la actualidad y que cubren necesidades semejantes. Del mismo modo, analizaremos sus ventajas y desventajas, y desarrollaremos una comparación con respecto a nuestro Trabajo de Fin de Grado.

- **Capítulo 3. Diseño y Análisis**

Analizaremos el diseño de nuestra aplicación.

- **Capítulo 4. Desarrollo**

Describiremos detalladamente cómo se ha desarrollado la plataforma web, destacando las implementaciones más relevantes y explicando cómo se han llevado a cabo.

- **Capítulo 5. Integración, pruebas y resultados**

Explicaremos cómo integrar la aplicación en un entorno real, así como las pruebas unitarias y funcionales que se crearon.

- **Capítulo 6. Conclusiones y trabajo futuro**

En el capítulo final, especificaremos las conclusiones y propondremos desarrollos y mejoras para el futuro.

- **Glosario**

En este apartado indicaremos los tecnicismos empleados y explicaremos el significado de determinadas palabras técnicas. Capítulo 4 (Desarrollo): Este bloque describirá detalladamente cómo se ha desarrollado la plataforma web destacando las implementaciones más importantes y explicando cómo se han llevado a cabo.

- **Anexos**

Adjuntaremos el manual del usuario, fragmentos de código y capturas de pantalla de las aplicaciones.

2 Estado del arte

2.1 Funcionamiento de Blockchain

Mediante el sistema de Blockchain conseguimos analizar multitud de elementos, difíciles de falsificar y rastreables en el tiempo, que cambian y se actualizan de manera constante. Es decir, podemos reparar en que Blockchain es una infraestructura abierta que acapara: diversidad de activos, historial de custodia, propiedad y ubicación de activos, certificados, contratos, objetos físicos e, incluso, información personal identificable. [8]

Sin embargo, es inviable estimar que Blockchain pueda ser la respuesta a todas nuestras preguntas. Es cierto que este método está suficientemente cualificado para solventar grandes problemas del mundo, pero todavía ésta en desarrollo. [9]

Actualmente, existen muchos expertos trabajando con tecnología Blockchain (instituciones financieras, compañías tecnológicas, Startup-Ups, universidades…), ya que este sistema no sólo ha engendrado una revolución económica, sino que en sí es una innovación en información. [10]

El sistema de Blockchain se ha transformado en una institución tecnológica que sobrepasa en numerosos aspectos a las instituciones tradicionales. Su principal característica es ser una institución descentralizada con capacidad tecnológica de instaurar registros de interacciones personales e intercambios monetarios, entre otros. [12]

Los cimientos sobre los que se sustenta el sistema de Blockchain son criptografías elementales como: funciones Hash, criptografía de clave pública, firma digital y ECDSA (Ellyptic Curve Digital Signature Algorithm). [7]

2.2 Tipos de Blockchain

En un principio, Blockchain fue planteado como un sistema público, en dónde todo el mundo podía participar. Por ejemplo, a través de Bitcoins se realizan transacciones públicas, seguras y transparentes, de manera inmediata y sin intermediarios. No obstante, más tarde, aparecieron sistemas de Blockchain privados e híbridos.

Sin embargo, existe otra clasificación, según generaciones:

La primera generación se asienta en la idea de crear un sistema de registro compartido en el cual se pueden contemplar las transacciones registradas.

En la segunda generación, que extiende la ideología del apartado anterior, se crea una red en la que se manejan criptomonedas y se guardan las relaciones del crédito de divisas determinadas por los usuarios.

Por último, en la tercera generación, se crean plataformas cuyo fin único es la creación de aplicaciones descentralizadas usando como infraestructura tecnológica las plataformas de las redes de la segunda generación. [1]

2.3 Bitcoin

Bitcoin es un activo sin naturaleza física que integra los fundamentos de la criptografía con el fin de alcanzar una economía segura, anónima y descentralizada. Compone los cimientos y el valor en torno al cual se ha implementado el sistema de Blockchain, por ello es importante comprenderlo.

2.4 Criptografía

La criptografía, ciencia que se encarga de analizar los algoritmos que ocultan información, es esencial para los sistemas de Blockchain.

En la década de los 90'emergieron los sistemas de clave asimétrica, los cuales disponen de dos claves: una privada y una pública. La clave pública permite tramitar mensajes cifrados, mientras que con la clave privada podemos rescatar dicho mensaje.

Ambas claves se caracterizan por ser invertibles, es decir, para desencriptar un mensaje es necesario conocer tanto la clave pública como la clave privada.

Existen dos tipos de sistemas de cifrado:

Los criptosistemas simétricos, o de clave privada, son aquellos que manejan, para cifrar y descifrar, una misma clave. Su principal inconveniente es que, para descifrar el mensaje, la clave debe ser conocida tanto por el emitente como por el receptor; esto lleva a cuestionarse cómo transferir la clave, de forma totalmente segura, al destinatario.

Los criptosistemas asimétricos, o de clave pública, son aquellos que emplean doble clave (privada y pública). La clave privada se emplea en el cifrado, mientras que la pública se maneja para el descifrado. En determinadas ocasiones, las claves son intercambiables, pero deben cumplir siempre una peculiaridad: es inviable descifrar la clave privada conociendo exclusivamente la clave pública.

Normalmente, en la práctica habitual, se emplea una combinación de estos dos tipos de criptosistemas. Por una parte, la criptografía asimétrica tiene una clara desventaja y es que es más cara que la criptografía simétrica. Sin embargo, por lo general, se emplea la criptografía asimétrica para codificar las claves simétricas y emitirlas, inclusive a través de canales inseguros. Por último, estas se codifican en mensajes más largos a través de algoritmos simétricos, que acostumbran a ser más eficientes.

El sistema criptográfico RSA se basa en el sistema ECDSA, aplicado por los sistemas de Blockchain. Las claves públicas y las claves privadas se calculan a partir de un número que se consigue como resultado de dos números primos grandes. Por ello, el sistema RSA se encarga de descifrar la dificultad que muestra la factorización de dichos números grandes.

2.5 Tierion

La empresa Tierion ha fundado una tecnología denominada Chainpoint. Esta permite revelar, con certeza absoluta, cuándo se crearon los datos de un documento y si este ha sido modificado en cualquier momento desde su estado original. [2]

Los servicios de salud, seguros y finanzas manejan grandes volúmenes de datos que necesitan ser salvaguardados. Por ello, es verdaderamente pesado y costoso comprobar la autenticidad de dicha información, alcanzando en muchos casos lo imposible. [2]

La empresa Tierion propone, mediante el uso de su plataforma, red Tierion, un resultado viable fundamentado en tecnología Blockchain, que permite verificar datos a gran escala.

La red Tierion permite vincular datos a una transacción en una cadena de bloques, de tal manera que cualquier persona puede comprobar la integridad y la marca de tiempo sin autorización.[15]

2.6 Factom

Factom es uno de los proyectos más sobresalientes que ha manado dentro de las aplicaciones de Blockchain. Esta firma ha anunciado un novedoso protocolo descentralizado levantado sobre la red de Bitcoin, el cual no trabaja mediante criptomonedas, sino a través de almacenamiento masivo de datos.[3]

Los datos que registran los usuarios en la red de Factom se codifican en funciones Hash; estos pueden ser consignados a su vez en bloques de mayor tamaño junto a otros producidos por el usuario primero. [16]

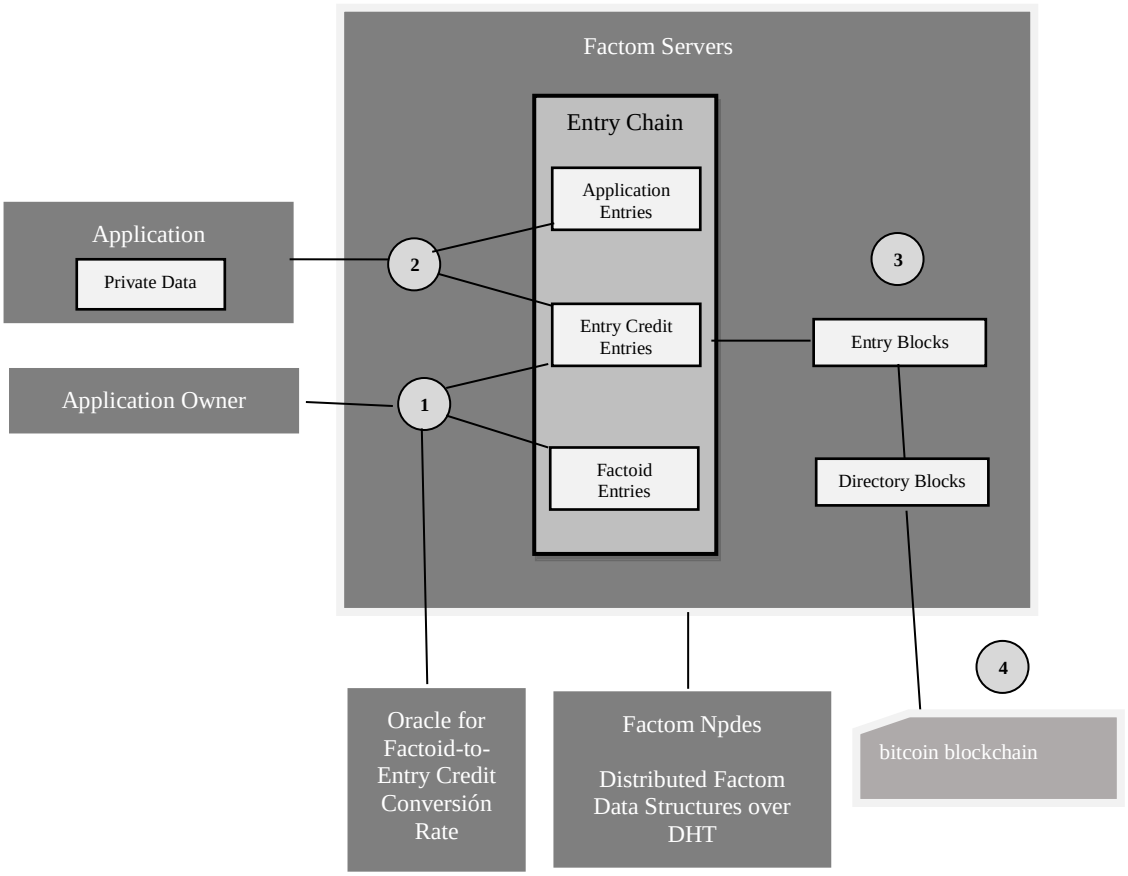

Figura 1: gráfico red de Factom

3 Diseño y Análisis

3.1 Árbol de Merkle

El árbol de Merkel es capaz de unir un gran número de datos proporcionando así un método de verificación segura y eficiente. Esta estructura de datos binaria, fundada en resultados de la función Hash, se divide en “hojas” y “nodos”. Las hojas representan un conjunto de bloques minados, y los nodos se encuentran etiquetados con un Hash de sus nodos hijos. [4].

El árbol de este Trabajo de Fin de Grado está formado por hojas que serán los Hash de los archivos que añadamos, mientras que los nodos, que no son hojas, se compondrán de la concatenación de los valores o las etiquetas de sus nodos hijos. [11]

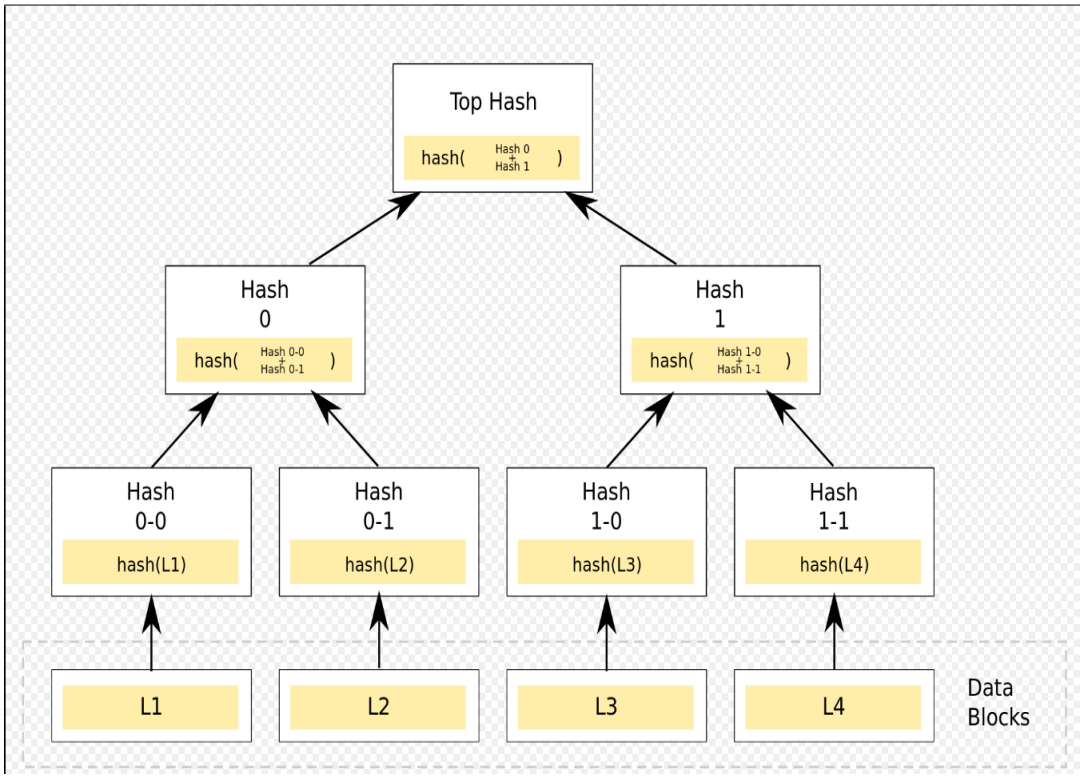

Figura 2: árbol de Merkle [4]

En el caso de las ramas impares, duplicaremos el nodo que quede sin hermano y, de este modo, sacaremos el nodo padre correspondiente.

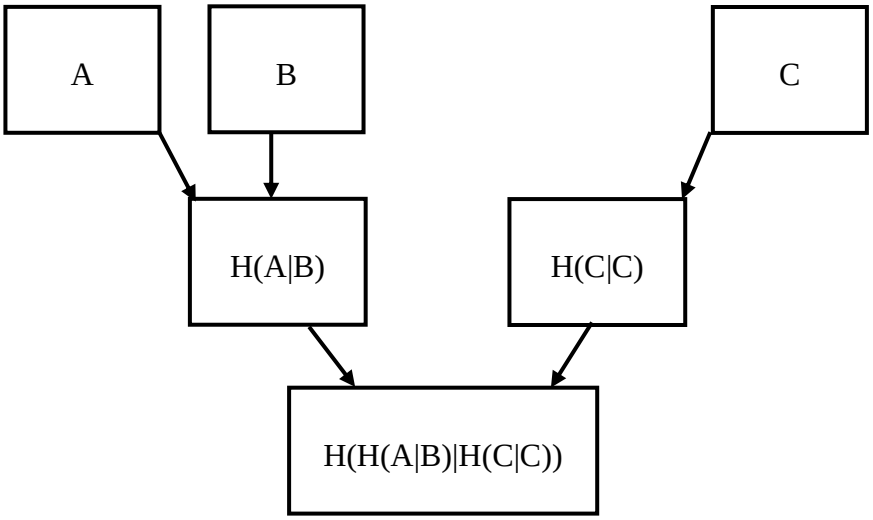

Figura 3: árbol de Merkle impar

3.2 Cifrado de archivos

Para el diseño del cifrado de archivos se ha decidido encriptar de forma simétrica mediante clave RSA [14]. El cifrado simétrico elegido es AES [6] y su cifrado se basa en matriz de estado. Tiene tamaños de 128, 192 y 256 bits de largo. El cifrado se realiza con el operador "or exclusivo" (XOR) y su operación se conoce como AddRoundKey.[13]

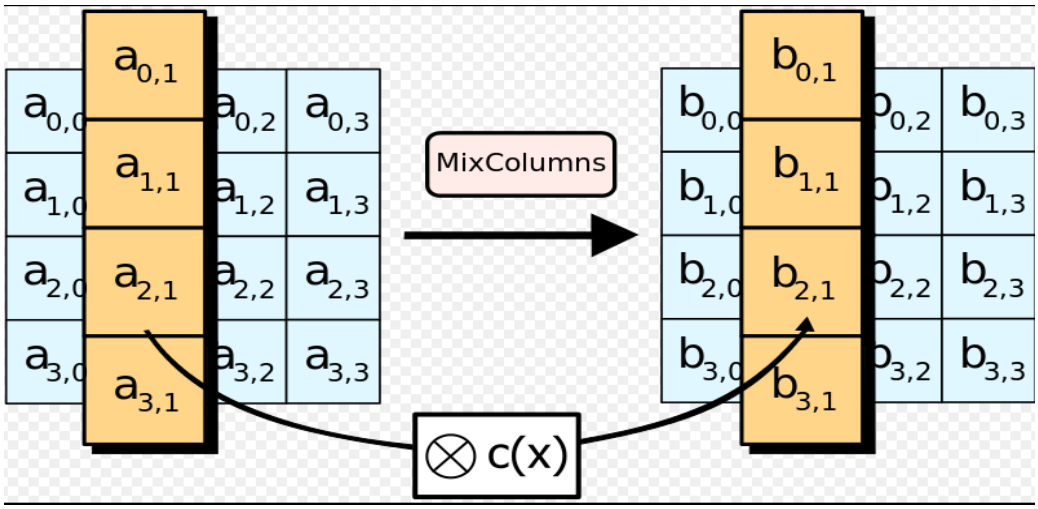

Figura 4: Cifrado Advanced Encryption Standard [6]

El cifrado para la clave de AES será el RSA (algoritmo de cifrado asimétrico), que se encripta mediante la clave pública. De esta manera, el receptor puede desencriptar con su clave privada la clave AES y desencriptar finalmente el archivo. [5]

3.3 Aplicación

El proyecto constará de dos páginas web y un servidor.

La primera página servirá para recibir el nombre y la clave pública asignada a la persona en cuestión.

En la segunda página incluiremos el botón donde se puede adjuntar el archivo, dos checkbox que permiten encriptar y/o guardar el archivo y un texto dónde se incluye el nombre de la persona en cuestión.

Una vez mandado el archivo, se enviará a nuestra base de datos. Al final del día se creará un bloque, mediante el árbol de Merkel que se remite a Blockchain y, si el bloque se añade correctamente se borrarán los archivos que no tengan que persistir. En el caso de que falle, se guardarán para probar de nuevo al día siguiente.

3.4 Análisis de Requisitos

3.4.1 Requisitos Funcionales

(RF1) Agregar usuario

Descripción: Se podrán agregar nuevos usuarios a la base de datos, añadiendo su clave pública para, posteriormente, ser utilizada.

Entrada: Se introduce el nombre y su clave pública.

Proceso: La aplicación comprueba que los datos se han introducido correctamente y lanza una consulta a la base de datos para crear el nuevo usuario.

Salida: La aplicación lanza un mensaje avisando de que el usuario se creó correctamente. Si existe el usuario, se lanzará un mensaje de error.

(RF2) Agregar archivos

Descripción: Se podrán agregar archivos.

Entrada: Se introduce el archivo pulsando el botón examinar.

Proceso: La aplicación añade el archivo de un formato aceptado.

Salida: La aplicación muestra el nombre del archivo, agregado, por pantalla.

(RF3) Enviar archivos

Descripción: Se enviará un archivo a un usuario seleccionado.

Entrada: Después de agregar un archivo, se introduce el nombre del usuario y se aprieta el botón enviar.

Proceso: La aplicación envía el archivo a la lista que formará él bloque.

Salida: La aplicación muestra que el archivo ha sido enviado correctamente. Si el usuario no existe, lanzará un mensaje de error.

(RF4) Enviar archivos guardando copia

Descripción: Se enviará un archivo a un usuario seleccionado y, además, se añadirá una copia en la aplicación.

Entrada: Después de agregar un archivo, se introduce el nombre del usuario, se maraca el check de guardar archivo y se aprieta el botón enviar.

Proceso: La aplicación comprueba que los datos se han introducido correctamente y lanza una consulta a la base de datos para añadir el archivo, enviando el archivo a la lista que formara él bloque.

Salida: La aplicación muestra que el archivo ha sido enviado correctamente. Si el usuario no existe, o si ha habido un error en la consulta a la base de datos, lanzará un mensaje de error.

(RF5) Encriptar archivos y enviar

Descripción: Se encriptará el archivo listo para enviar antes de ser enviado.

Entrada: Después de agregar un archivo, se introduce el nombre del usuario, se maraca el check de encriptar archivo y se aprieta el botón enviar.

Proceso: La aplicación comprueba que los datos se han introducido, después lanza una consulta a la base de datos para obtener la clave pública del usuario, encripta el archivo con una clave simétrica, generada de manera aleatoria, y a continuación, se encripta esta clave con la clave pública del usuario agregándose en la última línea del archivo. Al terminar, se envía el archivo a la lista que formará él bloque.

Salida: La aplicación muestra que el archivo ha sido enviado correctamente. Si el usuario no existe o, si ha habido un error en la consulta a la base de datos, lanzará un mensaje de error.

3.4.2 Requisitos no Funcionales

(RNF1) Requisitos de Disponibilidad

(RNF1.1) Al finalizar el intento de colocar el bloque, la aplicación deberá informar si el proceso funcionó correctamente, o si mantiene los archivos para el próximo intento.

(RNF2) Requisitos de Rendimiento

(RNF2.1) La aplicación mantiene un tiempo máximo, de 30 minutos, para intentar insertar el bloque.

(RNF3) Requisitos de Seguridad

(RNF3.1) La aplicación mantiene una copia de seguridad de los árboles subidos para poder comprobar en cualquier momento que los archivos están en dicho árbol, o bien si hubiera un error, poder volver a subir la raíz del árbol en un bloque.

(RNF3.2) Se realiza una encriptación del archivo con encriptación simétrica; la clave se encriptará con clave pública del receptor.

(RNF4) Requisitos de Plataformas

(RNF4.1) La aplicación web será accesible desde cualquier navegador web.

(RNF4.2) La base de datos utilizada será MySQL.

4 Desarrollo

La aplicación ha sido desarrollada en Python y la base de datos en MySQL. La web tendrá parte en HTML.

4.1 Aplicación Web

4.1.1 Conexión

La conexión a la base de datos MySQL se realizó mediante librerías ya creadas en Python dentro de flaskext.mysql. La conexión se realiza mediante la sentencia que podemos ver en el Anexo C – Fragmento de código 1.

El método msql.connect() establece una conexión por cliente y se cierra al terminar. Por último, y antes de entrar en el siguiente apartado, vamos a observar el código de una consulta simple en el Anexo C – Fragmento de código 2.

4.1.2 Seguridad

La aplicación, a priori, no presentaba fallas graves de seguridad, dado que no guarda contraseñas. No obstante, si acceden a la base de datos se podrán ver archivos encriptados con AES y RSA; los demás datos no contienen información comprometida.

Como explicamos en el diseño, los algoritmos utilizados para encriptar son RSA y AES.

$$c \equiv m^e \pmod{n}$$

Las claves AES las generaremos de manera aleatoria, por lo que no podremos conocer qué clave tendrá cada usuario. Esto se observa en el código del Anexo C – Fragmento de código 3.

4.1.3 Interfaz de usuario

En esta sección analizaremos la interfaz de usuario.

El usuario, en la primera página, tendrá un formulario para registrar el nombre de la persona al que le quiere enviar los datos, así como la clave pública.

Al enviar los datos, se guardarán en la base de datos, como podemos ver en el Anexo C–Fragmento de código 4. Del mismo modo, podremos observar el control de errores que se ha programado por si falla la inserción, o si el usuario no rellena todos los campos.

En la segunda página, el formulario será más completo, ya que constará de un combobox. De tal manera que, al enviar el código, tendrá más verificaciones de error. Ver Anexo Fragmento de código 5.

4.1.4 Servidor

El servidor es la parte más costosa, ya que aquí se crean los bloques. A las 12 de la noche salta una rutina (Anexo C – Fragmento de código 6). Esta utiliza nuestra clase ArbolMerkel(), en la que se saca el hash de todos los archivos siguiendo el método descrito anteriormente. Ver Anexo C – Fragmento de código 7. Una vez creado este árbol, cogemos el hash nodo raíz y creamos un bloque (Anexo C – Fragmento de código 8). Con el bloque creado empezaremos a minar (Anexo C – Fragmento de código 9, 10, 11, 12), si el minero consigue meter el bloque, el programa guarda la raíz y el json creado por el árbol, borrando su lista de archivos del día. Si el bloque no consigue entrar en la red de Blockchain, la lista de archivos se mantiene para que al día siguiente volver a crear el árbol.

Por otra parte, el servido, en local, tiene la página en funcionamiento para que cada vez que nos conectemos aparezca la página de inicio, así como el resto de las páginas web (Anexo C – Fragmento de código 13, 14, 15).

Más tarde, decidimos añadir los métodos para que el servidor pudiese comprobar que un archivo estaba en un árbol añadido. (Anexo C – Fragmento de código 16).

5 Integración, pruebas y resultados

En este apartado detallaremos cómo se ha llevado a cabo la integración en un entorno real. También describiremos cómo se realizó la fase de pruebas y cuál ha sido el resultado final del proyecto.

5.1 Pruebas funcionales

Para probar la aplicación, se realizaron un total de 3 pruebas. La primera comprobó que se recibieran bien los datos.

En la siguiente prueba comprobamos que los datos guardaban correctamente en la base de datos. En este punto, divisamos algunos fallos en los formatos del archivo. Este problema se solventó usando el formato apropiado para que se guardasen correctamente.

La siguiente prueba consistió en guardar varios archivos usando las opciones disponibles y comprobando que se agregasen correctamente a la cadena de bloques. El contratiempo que tuvimos en esta prueba fue que no se añadió adecuadamente el bloque a la cadena de bloques, teniendo que esperar a la siguiente franja horaria. Tras este tiempo de espera, pudimos resolver este fallo exitosamente.

5.2 Pruebas unitarias

La plataforma web está compuesta por dos formularios de datos, los cuales deben de ser rellenados para poder subir el archivo.

Se realizaron pruebas unitarias para comprobar por separado la funcionalidad de cada parte. A continuación, observaremos la siguiente tabla que recoge las pruebas unitarias que fueron realizadas.

Pruebas	Salida	Resultado
* Crear un Árbol de Merkle	* Sin errores	**OK**
* Obtener el diccionario	* Diccionario con todo el árbol	**OK**
* Obtener la raíz del árbol	* Hash Raíz del árbol	**OK**

Tabla 1: Pruebas unitarias del árbol

Pruebas	Salida	Resultado
* Crear un Servidor Blockchain	* Sin errores	**OK**
* Añadir nodo a la cadena de bloques	* Bloque se añade correctamente. * Mensaje de que no se unió.	**OK**
* Arrancar servidor	* Pagina principal del servidor	**OK**
* Rellenar el primer formulario	* Añade el usuario a la base de datos	**OK**
* Rellenar segundo formulario con la opción de encriptar	* Mensaje de confirmación	**OK**
* Rellenar segundo formulario con la opción de guardar.	* Se añade correctamente a la base de datos	**OK**
* Rellenar segundo formulario con la opción de guardar y encriptar.	* Se añade correctamente a la base de datos	**OK**
* Prueba de encriptación de archivo	*Archivo encriptado	**OK**
* Prueba de desencriptación de archivo	*Archivo desencriptado	**OK**

Tabla 2: Pruebas unitarias del servidor y del cliente Blockchain

5.3 Resultados del proyecto

El resultado del Trabajo de Fin de Grado es una página web dónde se recogen archivos encriptados, o sin encriptar, y se agrupan en un bloque, el cual se une a una cadena de bloques con el fin de ahorrar tiempo y dinero.

Todo el proyecto está disponible en GitHub, sin embargo, actualmente, se encuentra disponible con código privado. El repositorio que contiene el proyecto se encuentra en https://github.com/aitor77/proyecto.

6 Conclusiones y trabajo futuro

6.1 Conclusiones

Los objetivos planteados se han logrado al 100% e incluso hemos logrado más de lo previsto en la parte de usabilidad y cifrado, en particular hemos logrado crear una aplicación muy interesante para la seguridad de archivos, que proporciona costes asequibles comparado con otros métodos menos fiables.

Para lo cual, hemos desarrollado un sistema que permite, de manera segura, agregar una cuantiosa cantidad de archivos por bloque con el fin de devaluar el coste de cada transacción. Esto se consiguió mediante la implementar árboles de Merkel para lograr que todos los archivos que se reciban en un día se envíen como un solo bloque, lo que produce un ahorro de recursos notable.

Así mismo, hemos logrado añadir más seguridad al sistema, utilizando la clave pública para encriptar la clave simétrica, la cual se encarga, a su vez, de encriptar los documentos. Esto se hizo por medio de los sistemas de encriptación AES y RSA.

Finalmente, hemos diseñado una página web dónde el usuario puede, mediante opciones, enviar los archivos al bloque, encriptar el archivo y enviarlo al bloque, guardar el archivo y enviarlo al bloque o encriptar y guardar el archivo y enviarlo al bloque. Todo ello usando Flask, MySQL y Python, debido a las siguientes ventajas: lenguaje simplificado, rápido, flexible, ordenado y limpio.

Desde el punto de vista del reto personal: en un principio, el proyecto me parecía bastante complejo, pero, según iba avanzando, se me presentaba cada vez más accesible. No obstante, este proyecto ha evolucionado con creces de la idea inicial que tuve en un primer momento.

Teniendo en cuenta futuras versiones del proyecto, así como por cuestiones de mantenimiento y/o mejoras, se ha documentado el código para simplificar la tarea del desarrollador que lo retome.

6.2 Trabajo futuro

Para un futuro, será necesario mejorar la parte front, con el fin de que tenga un aspecto más bello. Igualmente, y según avance el sector, será necesario crear nuevas formas que acrecienten su optimización.

Asimismo, hemos dejado preparadas las funciones de la recuperación de los archivos de la cadena de bloques, así como las funciones de desencriptar por si cualquier desarrollador desea reanudar este proyecto en un futuro.

Referencias

[1] ¿Qué es la tecnología de contabilidad distribuida o blockchain? [Internet]. Criptonoticias. 2019 [cited 5 March 2019]. Available from: [1] https://www.criptonoticias.com/informacion/que-es-tecnologia-contabilidad-distribuida-blockchain/

[2] Simplificar la confianza [Internet]. Tierion. [cited 5 March 2019]. Available from: [2] https://tierion.com/

[3] Harmony Integrate [Internet]. Factom. [cited 5 March 2019]. Available from: [3] https://www.factom.com/

[4] Árbol de Merkle [Internet]. Wikipedia. 2018 [cited 5 March 2019]. Available from: [4] https://es.wikipedia.org/wiki/%C3%81rbol_de_Merkle

[5] RSA [Internet]. Wikipedia. 2019 [cited 5 March 2019]. Available from: [5] https://es.wikipedia.org/wiki/RSA

[6] Advanced Encryption Standard. 2019 [cited 5 March 2019]. Available from: [6] https://es.wikipedia.org/wiki/Advanced_Encryption_Standard

[7] Threshold-optimal DSA/ECDSA signatures and an application to Bitcoin wallet security [Internet]. 1st ed. New York: Princeton University; 2016 [cited 26 February 2019]. Available from: [7] https://eprint.iacr.org/2016/013.pdf

[8] ECDSA [Internet]. 1st ed. 2017 [cited 6 January 2019]. Available from: [8] http://libroblockchain.com/ecdsa/

[9] Para qué se está usando Blockchain más allá de bitcoin [Internet]. Xataka. [cited 15 December 2018]. Available from [9] https://www.xataka.com/empresas-y-economia/para-que-se-esta-usando-blockchain-mas-alla-de-bitcoin

[10] Qué es Blockchain y cómo funciona [Internet]. Dekalabs. 2019 [cited 1 June 2019]. Available from: [10] https://dekalabs.com/que-es-blockchain-y-como-funciona/

[11] Qué es el Árbol de Merkle [Internet]. Academy by Bit2me. [cited 14 June 2019]. Available from: [11] https://academy.bit2me.com/que-es-arbol-de-merkle/

[12] Fundamentos técnicos de blockchain -4. La cadena de bloques [Internet]. Steemit. 2018 [cited 14 May 2019]. Available from: [12] https://steemit.com/blockchain/@asuarez/fundamentos-tecnicos-de-blockchain-4-la-cadena-de-bloques

[13] AES [Internet]. NFON. 2019 [cited 30 May 2019]. Available from: [13] https://www.nfon.com/es/servicio/base-de-conocimiento/base-de-conocimiento-destacar/aes/

[14] ¿Qué es el cifrado RSA y cómo funciona? [Internet]. Comparitech. 2018 [cited 10 June 2019]. Available from: [14] https://www.comparitech.com/blog/information-security/rsa-encryption/

[15] ¿Qué es Tierion (TNT)? La big data de las criptos [Internet]. Bitcoin.es. 2019 [cited 12 June 2019]. Available from: [15] https://bitcoin.es/criptomonedas/que-es-tierion-tnt-la-big-data-de-las-criptos/

[16] Conoce a Factom, plataforma segura de almacenamiento en la blockchain de Bitcoin [Internet]. Criptonoticias. 2016 [cited 11 March 2019]. Available from: [16] https://www.criptonoticias.com/colecciones/factom-plataforma-segura-almacenamiento-blockchain-bitcoin/

Glosario

API	Application Programming Interface
HTML	HyperText Markup Language
SHA	Secure Hash Algorithm
AES	Advanced Encryption Standard
RSA	Rivest, Shamir y Adleman
ECDSA	Ellyptic Curve Digital Signature Algorithm

Anexos

A.1 Manual de instalación

Instrucciones de uso

iniciar el servidor de blockchain,

```
>>> python node_server.py
```

iniciar aplicacion,

```
>>> python run_app.py
```

La aplicación corre en esta dirección http: // localhost: 5000 .

Figura 5: Manual de instalación

A.2 Manual de Usuario

El primer paso es dirigirse a la url correspondiente del servidor

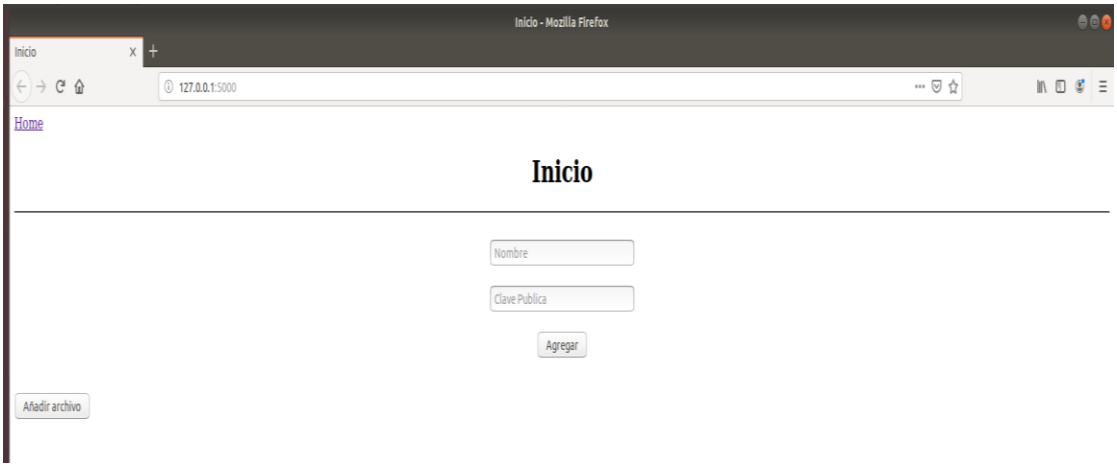

Figura 6: Página de inicio

Aquí se registrará el usuario al que se quiere enviar con su clave pública o podemos pasar añadir archivo. Si se rellena correctamente saltara un mensaje de confirmación.

Si damos a enviar sin datos aparecerá el siguiente mensaje de error.

Figura 7: Página de envíos

En esta segunda página indicaremos el nombre de la persona a la que se lo enviamos, seleccionaremos el archivo que queremos subir y si queremos que el archivo quede guardado, encriptado por nosotros o las dos cosas.

Una vez todo relleno clicar en el botón enviar y se procederá al envió. También podemos volver a la página de inicio mediante home.

Si se da a enviar sin nada saltara un error de que se deben rellenar todos los campos.

A.3 Fragmentos de código

En este anexo se adjuntan todas las capturas de pantalla utilizadas durante el trabajo que contienen fragmentos de código.

```
21
22    mysql = MySQL()
23
24    app.config['MYSQL_DATABASE_USER'] = 'root'
25    app.config['MYSQL_DATABASE_PASSWORD'] = '1234'
26    app.config['MYSQL_DATABASE_DB'] = 'blockchain'
27    app.config['MYSQL_DATABASE_HOST'] = 'localhost'
28    mysql.init_app(app)
29
30    conn = mysql.connect()
31    cursor = conn.cursor()
```

Fragmento de código 1: Conexión MySQL

```
insertCmd = "insert into temporal (id, archivo, dia) values ('" + indice + "','" + json.dumps(diccionario, ident=4) + "','"+ datetime.datetime.now
try:
    cursor.execute(insertCmd)
    conn.commit()
    print("arrbols guardado en la base")
except Exception as e:
    print("Hubo un problema al insertar los datos:" + str(e))
```

Fragmento de código 2: Insertar en MySQL

```
def encriptar(clave,archivo):
    if clave == None:
        print("El usuario no tiene clave publica asignada")
    else:

        # Cargamos la clave publica
        key = RSA.importKey(clave)
        # Instancia del cifrador asimetrico
        cipher_rsa = PKCS1_v1_5.new(key)
        # Generamos una clave para el cifrado simetrico
        aes_key = get_random_bytes(16)
        # Encriptamos la clave simetrico con la clave publica RSA
        enc_aes_key = cipher_rsa.encrypt(aes_key)

        # Abro el fichero lo copio en memoria y lo cierro
        f = open(archivo, 'rb+')
        d = f.read()
        f.close()

        # Encriptamos los datos con la clve simetrica
        cipher_aes = AES.new(aes_key, AES.MODE_EAX)
        d = cipher_aes.encrypt(d, ' ')
        f = open(archivo, 'wb+')
        f.write(d)
        # Aniadimos la clave simetrica codificada en la ultima linea del archivo
        f.write("\n")
        f.write(enc_aes_key)
        f.close()

        return f
```

Fragmento de código 3: Encriptación de archivos.

```
@app.route('/submit', methods=['POST'])
def submit_text():
    name = request.form["author"]
    key = request.form["key"]

    if key and name:
        insertCmd = "insert into usuarios (nombre, publicKey) values ('"+name+"','"+key+"');"
        try:
            cursor.execute(insertCmd)
            conn.commit()
            return json.dumps({'message':'Usuario registrado correctamente'})
        except Exception as e:
            print("Hubo un problema al insertar los datos:" + str(e))
            return json.dumps({'error': 'Hubo un error al intentar ingresar los datos'})
    else:
        return json.dumps({'error':'Tienen que rellenarse todos los campos'})
```

Fragmento de código 4: Envió de datos Formulario 1

```python
@app.route('/submitB', methods=['POST'])
def submit_archivo():
    name = request.form["author"]
    file = request.form["archivo"]
    save = request.form.get('guardar')
    encrip = request.form.get('encriptar')
    if name and file:
        selectCmd = "select publicKey from usuarios where (nombre='"+name+"');"
        insertCmd = "insert into datos (nombre, archivo) values ('"+name+"','"+file+"');"
        if save and encrip:
            cursor.execute(selectCmd)
            key = cursor.fetchone()
            try:
                cursor.execute(insertCmd)
                conn.commit()
            except Exception as e:
                print("Hubo un problema al insertar los datos:" + str(e))
                return json.dumps({'error': 'Hubo un error al intentar ingresar los datos'})
            b = encriptar(key,file)
            LISTA.append(b)

            return json.dumps({'message': 'guardar y encriptar'})

        elif save:
            try:
                cursor.execute(insertCmd)
                conn.commit()
            except Exception as e:
                print("Hubo un problema al insertar los datos:" + str(e))
                return json.dumps({'error': 'Hubo un error al intentar ingresar los datos'})
            LISTA.append(file)
            return json.dumps({'message':'guardar'})

        elif encrip:
            cursor.execute(selectCmd)
            key = cursor.fetchone()
            b = encriptar(key,file)
            LISTA.append(b)
            return json.dumps({'message': 'encriptar'})

        else:
            LISTA.append(file)
            return json.dumps({'message': 'ni guardar ni encriptar'})
    else:
        return json.dumps({'error': 'Tienen que rellenarse todos los campos'})
```

Fragmento de código 5: Envió de datos Formulario 2

```
def manda_bloque():

    indice = (indice +1)
    mi_arbol = ArbolMerkel()
    aux.append(LISTA)
    LISTA = []
    mi_arbol.lista = aux
    mi_arbol.crear_arbol()
    diccionario = mi_arbol.get_diccionario()

    insertCmd = "insert into temporal (id, archivo, dia) values ('" + indice + "','" + json.dumps(diccionario, ident=4) + "','"+ datetime.
    try:
        cursor.execute(insertCmd)
        conn.commit()
        print("arrbols guardado en la base")
    except Exception as e:
        print("Hubo un problema al insertar los datos:" + str(e))

    post_object = {
        'dia' : datetime.date.now(),
        'content': mi_arbol.get_raiz(),
    }

    # Submit a transaction
    new_tx_address = "{}/new_transaction".format(CONNECTED_NODE_ADDRESS)

    requests.post(new_tx_address,
                  json=post_object,
                  headers={'Content-type': 'application/json'})

    result = redirect( "{{node_address}}/mine")

    if result == 0:
        print ("El nodo no entro en la cadena de bloques")
    else:
        insertCmd = "insert into registro (idregistro, fecha, arbolmerkel, listaArbol, json) values ('" + indice + "','" + datetime.dateti
                                                                                                    ident=4) +"';"
        try:
            cursor.execute(insertCmd)
            conn.commit()
            print("El nodo ha entrado en la cadena de bloques correctamente")
            aux = []
        except Exception as e:
            print("Hubo un problema al insertar los datos:" + str(e))
```

Fragmento de código 6: Función que manda el bloque.

```
class ArbolMerkel:

    def __init__(self, lista=None):

        self.lista = lista
        self.diccionario = OrderedDict()

    def crear_arbol(self):

        lista = self.lista
        diccionario = self.diccionario
        aux = []

        for index in range(0, len(lista), 2):
            # elmento mas a la izquierda
            izquierda = lista[index]
            hash_izq = hashlib.sha256(izquierda)
            diccionario[lista[index]] = hash_izq.hexdigest()


             # elemento a la derecha del que esta el mas a la izquierda
            if index+1 != len(lista):
               izq_derecha = lista[index+1]
               hash_der = hashlib.sha256(izq_derecha)
               diccionario[lista[index + 1]] = hash_der.hexdigest()
               aux.append(hash_izq.hexdigest() + hash_der.hexdigest())

            else:  # si no hay duplicamos la izquierda
                aux.append(hash_izq.hexdigest() + hash_izq.hexdigest())




        # ajustamos la variable y lo volvemos a lanzar hasta llegar a la raiz
        if len(lista) != 1:
            self.lista = aux
            self.diccionario = diccionario

            self.crear_arbol()

    def get_diccionario(self):
        return self.diccionario

    def get_raiz(self):
        ultima = self.diccionario.keys()[-1]
        return self.diccionario[ultima]
```

Fragmento de código 7: Clase Árbol de Merkle

```
class Block:
    def __init__(self, index, transactions, timestamp, previous_hash):
        self.index = index
        self.transactions = transactions
        self.timestamp = timestamp
        self.previous_hash = previous_hash
        self.nonce = 0

    def compute_hash(self):

        block_string = json.dumps(self.__dict__, sort_keys=True)
        return sha256(block_string.encode()).hexdigest()
```

Fragmento de código 8: Bloque

```
94
95      def mine(self):
96
97          if not self.unconfirmed_transactions:
98              return False
99
100         last_block = self.last_block
101
102         new_block = Block(index=last_block.index + 1,
103                           transactions=self.unconfirmed_transactions,
104                           timestamp=time.time(),
105                           previous_hash=last_block.hash)
106
107         proof = self.proof_of_work(new_block)
108         self.add_block(new_block, proof)
109
110         self.unconfirmed_transactions = []
111         # announce it to the network
112         announce_new_block(new_block)
113         return new_block.index
```

Fragmento de código 9: Minero

```
def proof_of_work(self, block):

    block.nonce = 0

    computed_hash = block.compute_hash()
    while not computed_hash.startswith('0' * Blockchain.difficulty):
        block.nonce += 1
        computed_hash = block.compute_hash()

    return computed_hash
```

Fragmento de código 10: Prueba de trabajo

```
  @property
  def last_block(self):
      return self.chain[-1]

  def add_block(self, block, proof):

      previous_hash = self.last_block.hash

      if previous_hash != block.previous_hash:
          return False

      if not Blockchain.is_valid_proof(block, proof):
          return False

      block.hash = proof
      self.chain.append(block)
      return True
```

Fragmento de código 11: Añadir bloque

```
@classmethod
def is_valid_proof(cls, block, block_hash):

    return (block_hash.startswith('0' * Blockchain.difficulty) and
            block_hash == block.compute_hash())
```

Fragmento de código 12: Es valida la prueba

```
<html>
  <head>
    <title>{{ title }}</title>

  </head>
  <body>
    <div><a href="/">Home</a></div>
    <center><h1>{{ title }}</h1></center>
    <hr>
    {% with messages = get_flashed_messages() %}
    {% if messages %}
    <ul>
    {% for message in messages %}
        <li>{{ message }} </li>
    {% endfor %}
    </ul>
    {% endif %}
    {% endwith %}
    {% block content %}{% endblock %}
  </body>
</html>
```

Fragmento de código 13: Base.html

```
<!-- extend base layout -->
{% extends "base.html" %}

{% block content %}
      <br>

      <center>
      <form action="/submit" id="textform" method="post">
          <input type="text" name="author" placeholder="Nombre">
              <br><br>
              <input type="text" name="key" placeholder="Clave Publica">
              <br><br>
          <input type="submit" value="Agregar">


      </form>
      </center>

      <br>
      <a href="/envios" target="_blank"><button>Añadir archivo</button></a>

{% endblock %}
```

Fragmento de código 14: Index.html

```
<!-- extend base layout -->
{% extends "base.html" %}

{% block content %}
      <br>

      <center>
      <form action="/submit8" id="textform" method="post">
          <input type="text" name="author" placeholder="Nombre">
              <br><br>
              <input type="file" name="archivo" placeholder="Añadir fichero">
              <br>
              <label><input type="checkbox" name="encriptar" value="encriptar_check"> Encriptar archivo</label><br>
              <label><input type="checkbox" name="guardar" value="guardar_check"> Guardar archivo</label> <br>
              <br>
              <input type="submit" value="Enviar">


      </form>
      </center>

      <br>
      <div style="margin: 20px;">

    {% for post in posts %}
    <div class="post_box">
         <div class="post_box-header">
            <div class="post_box-options"><button class="option-btn">Reply</button></div>
            <div style="background: rgb(0, 97, 146) none repeat scroll 0% 0%; box-shadow: rgb(0, 97, 146) 0px 0px 0px 2px;" class="post_b
            <div class="name-header">{{post.author}}</div>
            <div class="post_box-subtitle"> Posted at <i>{{readable_time(post.timestamp)}}</i></div>
         </div>
         <div>
            <div class="post_box-body">
               <p>{{post.content}}</p>
            </div>
         </div>
      </div>
      {% endfor %}
```

Fragmento de código 15: Envios.html

```
def comprobar_archivo(self,archivo,idRegistroArbol):
    hoja = hashlib.sha256(archivo)
    selectCmd = "select hojascol from hojas Where hojascol ="+hoja+" and
idRegistro = "+ idRegistroArbol+"');"
    cursor.execute(selectCmd)
    key = cursor.fetchone()
    if key is None :
        return True
    else:
        return False
```

Fragmento de código 16: Comprobar archivos